AIAA-2007-3834

On Strand Grids for Complex Flows

Robert L. Meakin
U.S. Army, Research, Development, and Engineering Command
AMRDEC (AFDD), Moffett Field, CA

Andrew M. Wissink
ELORET Corp., Moffett Field, CA

William M. Chan and Shishir A. Pandya
NASA Ames Research Center, Moffett Field, CA

Jayanaryanan Sitaraman
National Institute of Aerospace, Hampton, VA

18th AIAA Computational Fluid Dynamics Conference

June 25 - 28, 2007 / Miami, Florida

On Strand Grids for Complex Flows

Robert L. Meakin[1]
US Army Research, Development, and Engineering Command
Aviation and Missile Research, Development, and Engineering Center
Aeroflightdynamics Directorate, Moffett Field, CA 94035-1000

Andrew M. Wissink[2]
ELORET Corp., Moffett Field, CA 94035-1000

William M. Chan[3] and Shishir A. Pandya[4]
NASA Ames Research Center, Moffett Field, CA 94035-1000

Jayanaryanan Sitaraman[5]
National Institute of Aerospace, Hampton, VA 23666

The need for highly automated and computationally efficient tools for high fidelity simulation of complex flow fields is recognized. A discretization paradigm that holds significant advantages relative to these needs is described. Problem domains are categorized into near- and off-body partitions. Strand grid technology is applied to near-body partitions, while block-structured Cartesian AMR (Adaptive Mesh Refinement) is applied to the off-body partition. The computational advantages and degrees of automation derivable from the approach are reviewed. A set of software tools that have been developed for grid generation and flow solution using both strand grid and block-structured Cartesian AMR are presented. Demonstration of strand grid technology is provided via time-dependent flow simulations and comparison with experimental data. The degree to which strand grid technology expands the spectrum of problems that can be considered via high performance computing is also considered.

I. Introduction

THE demand for automated high-fidelity multi-physics modeling and simulation capability is fueled by powerful economic, political, and very practical constraints. An aging infrastructure for traditional methods of design data acquisition represents a staggering cost of refurbishment and long-term maintenance. In a time of war, environmental catastrophes, and aging populations, the politics of such a path is untenable. Globalization of the world marketplace has created fierce competition in all areas of technology, and represents very formidable challenges in terms of national economy and defense. These are some of the factors driving significant paradigm shifts required for product concept formulation, design, development, and production. Clearly, High Performance Computing (HPC) has a key role to play in how these realities are addressed. The present article presents a new paradigm for high-fidelity modeling and simulation that has potential for very high degrees of automation and that is design cycle friendly.

The paradigm discussed here addresses the method of discretization for high-fidelity simulation modeling of fluid/aero dynamic systems and how it lends itself to general multi-physics applications involving the interaction of fluids and structures. The article states the need for the technology and explores the advantages and risks associated with the proposed paradigm. Details of grid generation, flow solver, and domain connectivity modules required for the paradigm are described. Feasibility of the approach is demonstrated with a three-dimensional unsteady viscous flow application.

[1] Senior Staff Scientist, Senior Member AIAA.
[2] Senior Research Scientist, Member AIAA.
[3] Computer Scientist, Senior Member AIAA.
[4] Aerospace Engineer, Member AIAA.
[5] Research Scientist, Member AIAA.

II. Methods

Current methods available for high fidelity simulation modeling are very powerful. However, the constraints of our time pose large demands on the science, engineering, and HPC communities. HPC relevance in the marketplace is dependent on automation, physical accuracy, and timeliness of data. This means that design engineers, as opposed to software developers or experts, can easily define case conditions and robustly generate all required simulation initiation materials, including all grid related data. Further, this means that design data generated through HPC must be of sufficient physical accuracy and span of design space for decisions to be reliably made. Finally, this means that the design data generated is available when it is needed. In spite of the truly remarkable advances that have been made in HPC and the computational sciences, current high-fidelity flow simulation software tools fall significantly short of what is required for relevance in a design environment. The level of expertise required to successfully use current grid generation and flow solver tools is too high and the slope of the associated learning curves too steep. Although current simulation software can predict flow physics of sufficient accuracy for design purposes for many flow regimes, there are situations that are problematic. Practical means of *a-postiori* data quality certification simply do not exist.

The paradigm and methods described in this article directly address the shortcomings in current technology in regards to automation and timeliness of data. At least, the material holds significant potential to positively address the needs. Additionally, the paradigm has the potential of expanding the range of problems that can be considered via HPC.

A. Spatial Partitioning

Cartesian methods of grid generation can be made fully automatic and completely robust for arbitrary geometric configurations. A variety of software packages exist that demonstrate that this is so. These include both unstructured Cartesian methods[1-3] in which cell subdivision is used to adapt to problem geometry, and block-structured approaches[4,5] which likewise adapt to problem geometry, but retain data structure in a block-wise sense. Of course, the reason Cartesian methods cannot be used exclusively to satisfy HPC requirements for automation is the length scales associated with viscous flow. Cartesian methods require isotropic refinement to accommodate arbitrarily complex boundary surfaces. Fluid/aerodynamic interactions with solid surfaces result in viscous boundary layers that require spacing too fine to be accommodated isotropically.

Table: Block-Structured Cartesian Grids – Integration to the Wall

Spacing Requirement	Minimum Spacing	Total Points Required
"Transitional"	$1.0\text{x}10^{-02}$	$1.86 \times 10^{+06}$
Inviscid	5.0×10^{-03}	$5.89 \times 10^{+06}$
Viscous	1.0×10^{-06}	$8.43 \times 10^{+12}$

The table above indicates the total number of Cartesian grid points required to integrate to the wall for the reentry vehicle illustrated in Figure 1. The data corresponds to block-structured Cartesian AMR with an amplification factor of 2 between refinement levels, a flight Reynolds number of $2.5\text{x}10^{+6}$, for a set of 3 terminal refinement states; transitional, inviscid, and viscous. In this instance, "transitional" spacing is simply a state of refinement that is typical of that appropriate away from solid surfaces and outside the viscous boundary layer. Viscous spacing is that sufficient for Reynold's Averaged Navier Stokes (RANS) or Detached Eddy Simulation (DES) applications. Exclusive use of Cartesian grids to solve inviscid flow problems for arbitrarily complex domains is practical on current HPC systems and can meet the degree of automation required for a design environment. However, as suggested by the data in the table above, exclusive use of Cartesian grids for viscous flow is many orders of magnitude beyond the capacity of current HPC systems.

The need for automation for general flow situations, including viscous flows, demands an alternative strategy. A method of spatial partitioning[6] is fundamental to the discretization paradigm described in this article. Here, problem domains are categorized into "near-body" and "off-body" partitions. The near-body partition is defined to include the surface geometry of all bodies being considered and the volume of space extending a short distance away from the respective surfaces (see

Figure 2). The discretization of the near-body partitions must resolve the viscous boundary layer and other flow features that evolve within this subset of the overall problem domain. The off-body is defined to include all other portions of the domain. Cartesian methods are ideal for the off-body domain as defined here. In this sense, near-body grids simply serve as a means to transition resolution capacity from the required viscous spacing at solid surfaces to positions in the field where it is practical for Cartesian methods to provide automatic coverage.

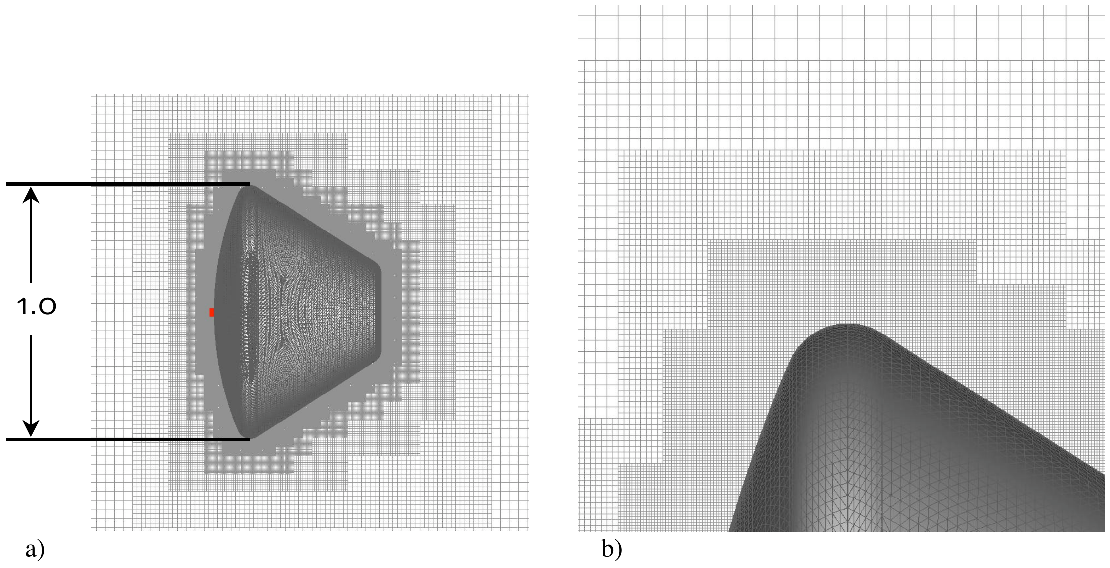


Figure 1. Block-structured AMR grid for a reentry capsule. a) Configuration (minimum spacing = 2.25×10^{-03}, inviscid). b) Close-up.

The novel contribution of this article is the introduction of "strand grids" to fill this transitionary role. Strand grid technology accepts surface geometry discretizations with n-sided polygons. Triangular elements are taken as the default in this article, but the algorithms described are equally applicable to n-sided elements. Typically, one strand is generated for each surface vertex. All strands are line segments with the origin, or root, positioned at the respective surface element vertex and aligned with a corresponding unit direction vector. Except for their roots and direction vectors, all strands are identical – sharing the same number and distribution of points along the strand. Spacing along each strand ranges from viscous at the root to "transitional" at the tip.

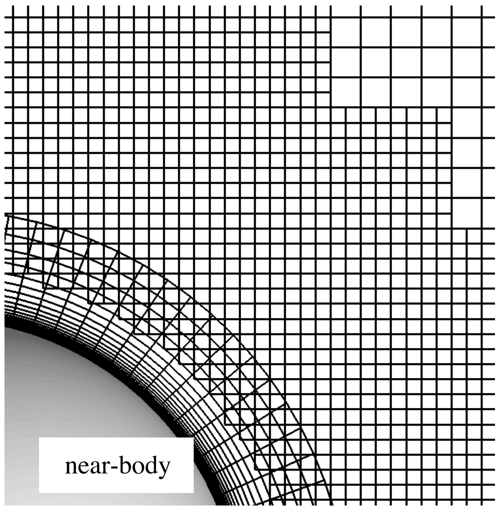
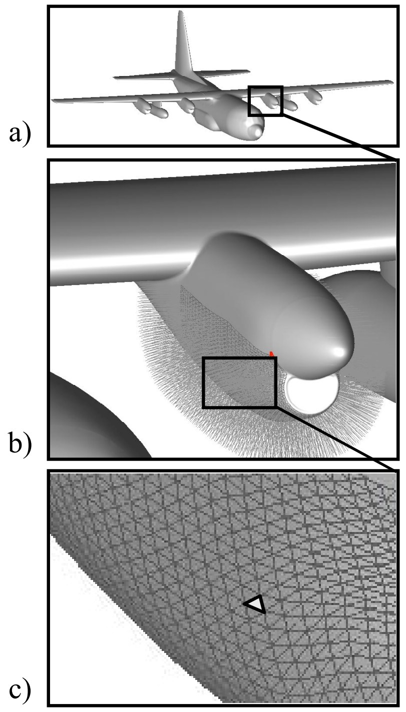


Figure 2. Sphere near-body grid overlaps off-body grids.

Consider the example given in Figure 3. A surface definition of the KC-130 aircraft is indicated in Figure 3a. Close-ups of a section of the left inboard engine provide views of selected strands and surface elements used in the discretization. Figure 3d is a strand definition sketch showing unit direction vectors, a strand root, tip, and "clipping-index." The solid triangular "bowtie" symbol used in Figure 3d is used consistently in figures of this article to designate clipping-index position along a strand. Clipping-indices are discussed further in Section II.B. For now, it is sufficient to note that all points between the strand root and point indicated by the clipping-index are valid field points. All points beyond the clipping-index to the tip, inclusive, are not used in the flow solution process except, as needed for assigning inter-grid boundary conditions. The motives for using strands defined in this way are multiple.

i. Strands satisfy the required transitionary role described previously.
ii. They provide possibility for fully automated grid generation (see Section II.B).
iii. They retain data structure in the viscous direction, facilitating efficient solution algorithms and time-integration methods (see Section II.C).
iv. They require virtually no memory for volume grid related data.



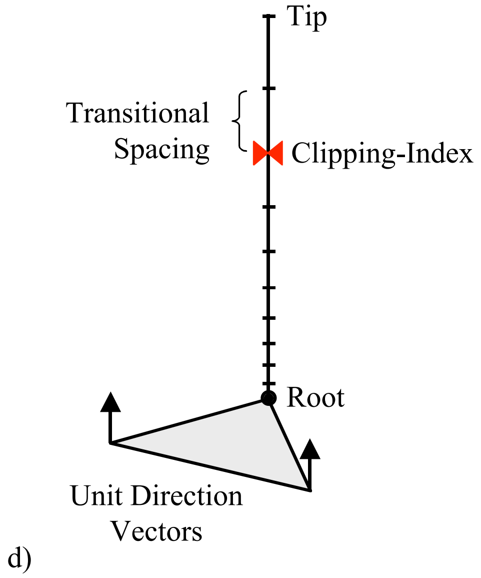


Figure 3. Strand Grids. a) KC-130 surface geometry, b) close-up of strand grids, c) close-up of surface triangulation. d) Strand definition sketch.

Strand grid technology collapses memory requirements for near-body domain partitions from volume data to surface data only. Non-surface grid related data that may be required during a simulation is trivially computed. It is easy to undervalue the advantage of minimal memory requirements for grid related data. After all, memory is relatively cheap. The advantage is realized in context of a design environment with HPC systems. Maximum solution throughput is obtained by allocating the minimum number of processors that can accommodate a given problem in core memory. Multiple cases can then be run simultaneously, realizing perfect parallelism.

The coupling of near-body strand grids with block-structured Cartesian AMR in the off-body, as indicated in Figure1a, represents very powerful technology. In addition to the fact the Cartesian AMR methods can be made fully automatic and robust, there is also no memory requirement for any grid related data. Each grid component is completely defined by the coordinates of the block diagonal (6 REALs), the resolution capacity of the component (1 REAL), and possibly an integer array defining point status. In total, 7 REAL numbers must be stored for each AMR block, which may be 100's or 1,000's for complex applications, as opposed to 10's to 100's of millions of REAL numbers that would be required for the full volume grids. In addition, the off-body flow solution process can fully exploit structured data, minimizing FLOP count requirements per flow solver iteration, efficiently using cache, and maximizing algorithm simplicity to implement powerful numerical methods such as multigrid sequencing, high-order methods, etc. Such methods are most efficient and accurate when implemented for uniform Cartesian grids, which are native to block-structured AMR. The present approach relies on block-structured AMR for domain coverage from the far-field boundaries, telescoping down to the extent of near-body strand grid coverage capacity identified by the set of strand clipping indices.

B. Near-Body Strand Grids

In its most general form, strand grid technology is independent of surface grid topology. That is, a direction vector and a 1-D grid point distribution can be assigned to each vertex on the surface, irrespective of how the surface grid points are connected. This means that the surface grid can be made up of structured abutting or overlapping patches, or a hybrid unstructured mixture of arbitrary polygons. The unstructured surface tessellation is attractive here because a tessellation that is sufficient for geometric definition can be easily and automatically generated from CAD geometry for complex configurations.

In the present article, an unstructured surface triangulation is used as the starting point for strand grid development. The procedure described is easily extensible to other types of unstructured surface tessellations. Starting from a native CAD solid model of the geometry, the CAPRI interface is used to create a surface triangulation automatically.[7,8] User control is available to adjust the triangle cell size, the maximum dihedral angle between normals of adjacent triangles, and the maximum cell centroid deviation from the CAD surface. The resulting triangulation is sufficient to describe the geometry. However, it is typically inadequate for near-body volume grid generation and flow field resolution since it lacks grid point clustering at sharp convex edges and a smooth cell size transition from small to large cells (see Figure 4). Here, it is assumed that a surface triangulation with such properties can subsequently be obtained using existing unstructured grid generation algorithms and software such as those described by Marcum.[9]

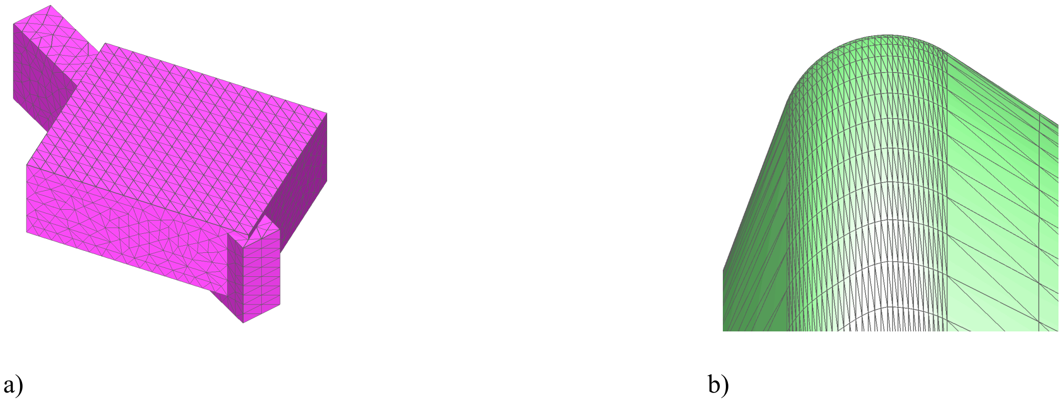

a) b)

Figure 4. Automatically generated surface triangulations from CAPRI. (a) No grid clustering at sharp edges. (b) Unsmooth cell size transition.

1. Strand Grid Generation

Given a closed surface triangulation with appropriate grid point distribution, strand grids are generated trivially by determining a direction vector at each surface vertex, and using a 1-D wall-normal point distribution function. Each surface vertex is assigned a different direction vector, but all strands share the same 1-D grid point distribution. A first cut at the direction vector at a vertex is computed by averaging the face normals of the triangles that share the vertex. This is simply one of several ways for computing the surface normal at a vertex. There are also many ways to define the 1-D wall-normal point distribution function. Here, the hyperbolic tangent stretching function is used where the spacing at the surface, spacing at the outer boundary, and a maximum allowed stretching ratio is prescribed. A default maximum stretching ratio of 1.2 is used to limit grid induced truncation error. The initial spacing away from the wall is typically computed as a function of the Reynolds number (for viscous flow simulations), to produce y+ values of approximately 1, or less. The optional outer boundary spacing is set to be the "transitional" spacing referred to earlier in Section II-A.

2. Strand Direction Vector Smoothing

After computing an initial set of direction vectors for all strands, it is immediately clear that certain regions of the near-body domain are inadequately resolved. This is particularly true in regions above convex features of the surface geometry (see Figure 5a). Such convex features include high curvature regions such as leading edges of wings, as well as sharp convex edges of the geometry. Volume resolution above these features can be improved by direction vector smoothing, or root-bending, in the vicinity of the convex features (see Figure 5b). This is achieved by averaging each vector with its neighbors. Currently, the amount of direction vector smoothing is an adjustable parameter. However, ways to set this parameter automatically are being investigated to minimize loss of orthogonality at the surface. An example of the grid quality realized with the current implementation of direction vector smoothing is illustrated on cut planes near the base of a dart geometry in Figure 6.

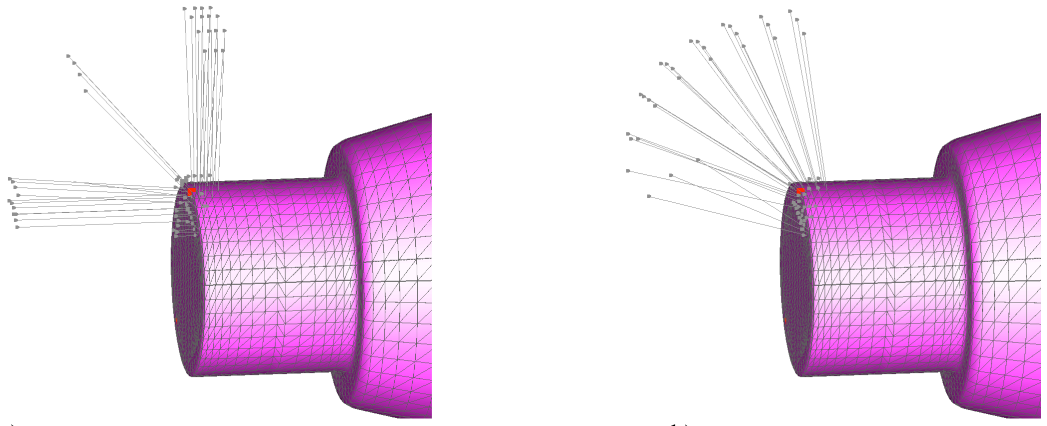


Figure 5. Strand grid at convex region using (a) default strand direction vectors based on local surface normals, (b) smoothed strand direction vectors.

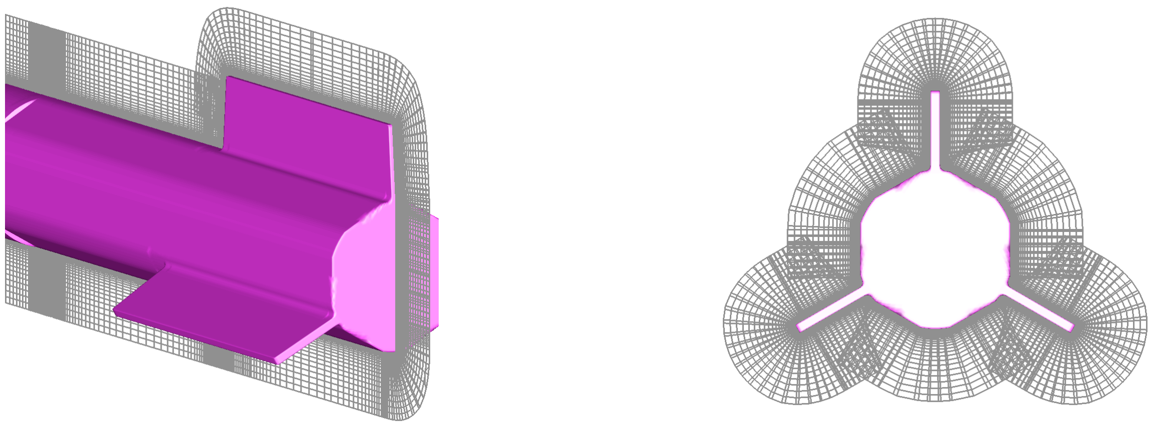


Figure 6. Cut plane showing smoothed strands on dart geometry. (a) Constant y cut. (b) Constant x cut.

It must be emphasized here that abrupt changes in surface point distributions are amplified in strand volume grids. Accordingly, for good volume resolution, especially at convex regions, the surface grid must have sufficient grid point clustering. Otherwise, direction vector smoothing alone is inadequate to provide the proper resolution in certain situations. In Figure 7, for example, more surface grid points are needed at the wing trailing edge (indicated by the arrow) to provide adequate volume coverage.

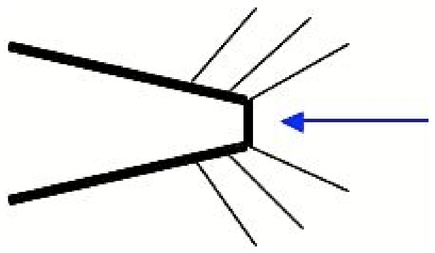


Figure 7. Inadequate surface grid point clustering near a convex corner that results in poor volume space resolution.

3. Strand Clipping-Index

Aside from situations correctable through direction vector smoothing, only one type of strand volume grid anomoly warrants consideration: negative volume prism cells. The volume of space above a triangular cell on the surface can be thought of as a stack of triangular prism cells defined by the three strands that emanate from the surface triangle. In the vicinity of certain surface features, such as sharp concave and convex edges, and saddle points, the three strand direction vectors for a corresponding prism stack may cross and result in negative volume prism cells beyond the cross-over point (see Figure 8). Cross-over points are marked by the "clipping-index," along such strands. Points on a strand beyond the clipping-index are not used in flow computations. In this way, the strand clipping-index is used to remove all negative volume prism cells from the near-body domain. The clipping index, by definition, is NFRINGE points from the strand-tip or strand-point associated with the first negative volume prism in a stack. NFRINGE is a user specified parameter that indicates minimum grid overlap needed for inter-grid communication (default value is 2).

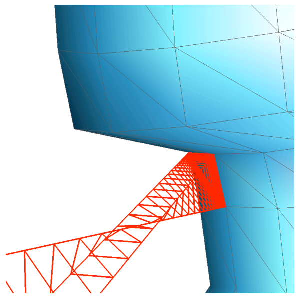


Figure 8. Negative prism cell volumes resulting from crossing strands in 3-D.

C. Strand Grid Flow Solver

The present near-body discretization strategy can be thought of as a structured grid in the strand-wise direction. Strands are made so that points along the strand are clustered close to the body surface. Spacing between points along the strand increase with distance from the surface. This structure facilitates natural use of implicit flow solution methods in the strand direction. Since this approximates the body-normal direction, a substantial reduction in CFL-limitations can be achieved. It is true that unstructured flow solvers can exploit line-implicit methods also, but only after significant processing to identify lines within the grids. In contrast, line structure is inherent to strand grids and represents a significant advantage.

As noted earlier, strand grid technology accepts surface geometry discretizations with *n*-sided polygons. Surface tessellation into triangular elements result in near-body volume grids comprised of triangular prism cells organized into structured stacks. Any unstructured flow solver that accommodates triangular prisms can be used as a strand grid flow solver. Such is the case with the examples considered in Section III, where NSU3D[10] is used as the near-body compute engine. Still, a solver that fully exploits the advantages inherent to strand grids should be far superior to present unstructured solvers in terms of memory requirements, and significantly more efficient in terms of required floating point operations. A future goal of the present work is to develop such a Navier-Stokes solver – one that is flexible, efficient, and accurate.

A hope of the author's is that fully automatic methods of surface geometry tessellation into quadrilateral elements with point distributions suitable for flow simulation will soon be a reality. If such is the case, the computational efficiencies inherent to fully block-structured data will also be realizable with strand-grid technology. The structure inherent in the strand direction, combined with structure easily derivable via agglomeration of surface quadrilateral elements, produce abutting block-structured grids. Here again, existing structured grid flow solvers can be employed to solve on agglomerated quadrilateral-element based strand grids. Yet, it is anticipated that such solvers can be optimized to exploit the advantages inherent to strand grids (viz., lower memory and FLOP requirements). Of course, the issues noted in Section II-B regarding direction vector smoothing, negative cell volumes, and clipping indices are independent of surface element type, and are resolved as described previously.

D. Off-Body Discretization Paradigm

Construction of the off-body grid system follows a traditional structured Cartesian AMR gridding paradigm.[11] Block-structured Cartesian AMR grids are formed from a hierarchy of nested refinement levels with each level composed of a union of logically-rectangular grid regions (Figure 9). All grid cells on a particular level have the same spacing, and the ratio of spacing between levels is generally a factor of two or four, although it is possible to use other refinement ratios. Simulation data are stored on patches in contiguous arrays that map directly to the grid points, without indirection. Levels are generated from coarsest to finest through a process where cells on the coarser level are marked for refinement, the marked cells are clustered into the logically-rectangular blocks, and these blocks form the patch boundaries on the finer level. A consequence of this approach is that finer level patch boundaries always align with coarse cell boundaries, which facilitates data communication between levels and numerical interpolation of data at the level boundaries.

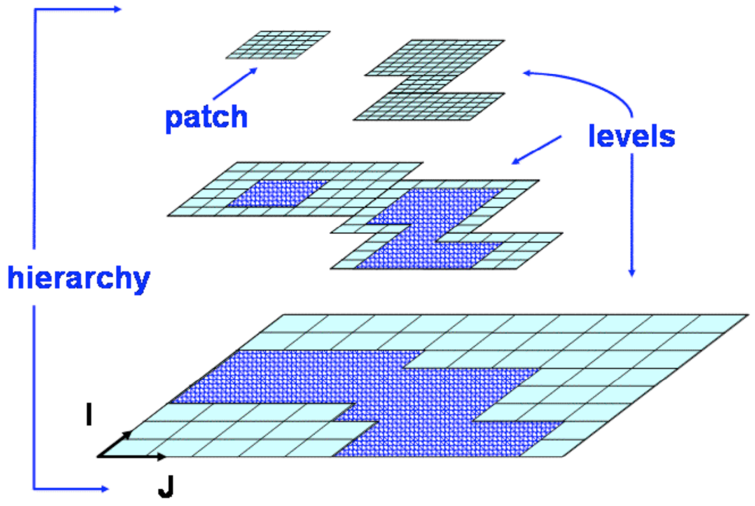


Figure 9. Block structured AMR grid system. A hierarchy of nested levels of refinement (each uniformly spaced level consisting of logically rectangular regions).

These off-body grid systems are constructed to match resolution associated with the clipping-indices of near-body strands. Accordingly, resolution compatibility at the intersection of off-body (block-structured Cartesian AMR) and near-body (strand) grids is always optimal. Given a set of strand clipping-indices, off-body AMR construction begins with a large coarse Cartesian grid that covers the entire domain, extending to the far-field boundaries. This coarse grid serves as the coarsest level in the AMR hierarchy. Cells that contain strand clipped-points are marked on the coarse level Cartesian grid. The marked cells are refined to construct blocks that form the next finer level. This procedure is repeated recursively to generate the nested system of grid levels.

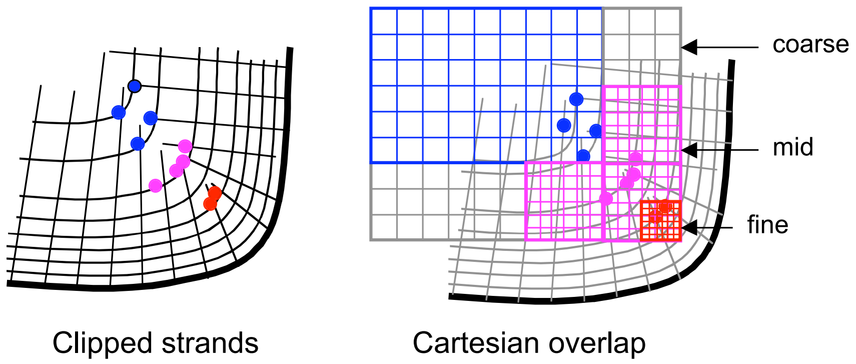


Figure 10. Refined grids that match spacing indicated by strand clipping-indices.

Comparable resolution in the overlap region of off-body Cartesian and near-body strand grids is achieved by considering the spacing between consecutive strand points measured from the clipping-index toward the strand-tip during the cell marking step. The Cartesian cell is only marked for refinement if the spacing of the cell is more than that indicated by the clipping-index. If it is less than or equal, the Cartesian grid has sufficient resolution in the overlap region to properly interpolate between the near- and off-body grid systems and no further refinement is necessary (see Figure 10). In this way, generation of the off-body

Cartesian grid system is completely automatic.

Figure 11 shows the Cartesian grid system generated for a strand representation of a wing-body configuration. The grid consists of 14 levels with most of the points residing on level 8. The spacing of the Cartesian grids at this level corresponds to the maximum transitional spacing (i.e., the maximum spacing available in the strand definition, see Figure 3d). For this case, a significant portion (about 95%) of the strands is satisfied with maximum transitional spacing. For those strands clipped nearer to the roots, Cartesian grids are automatically refined to the spacing indicated by the corresponding clipping-indices. Note that regions of very fine Cartesian grids are generated at the wing body junction.

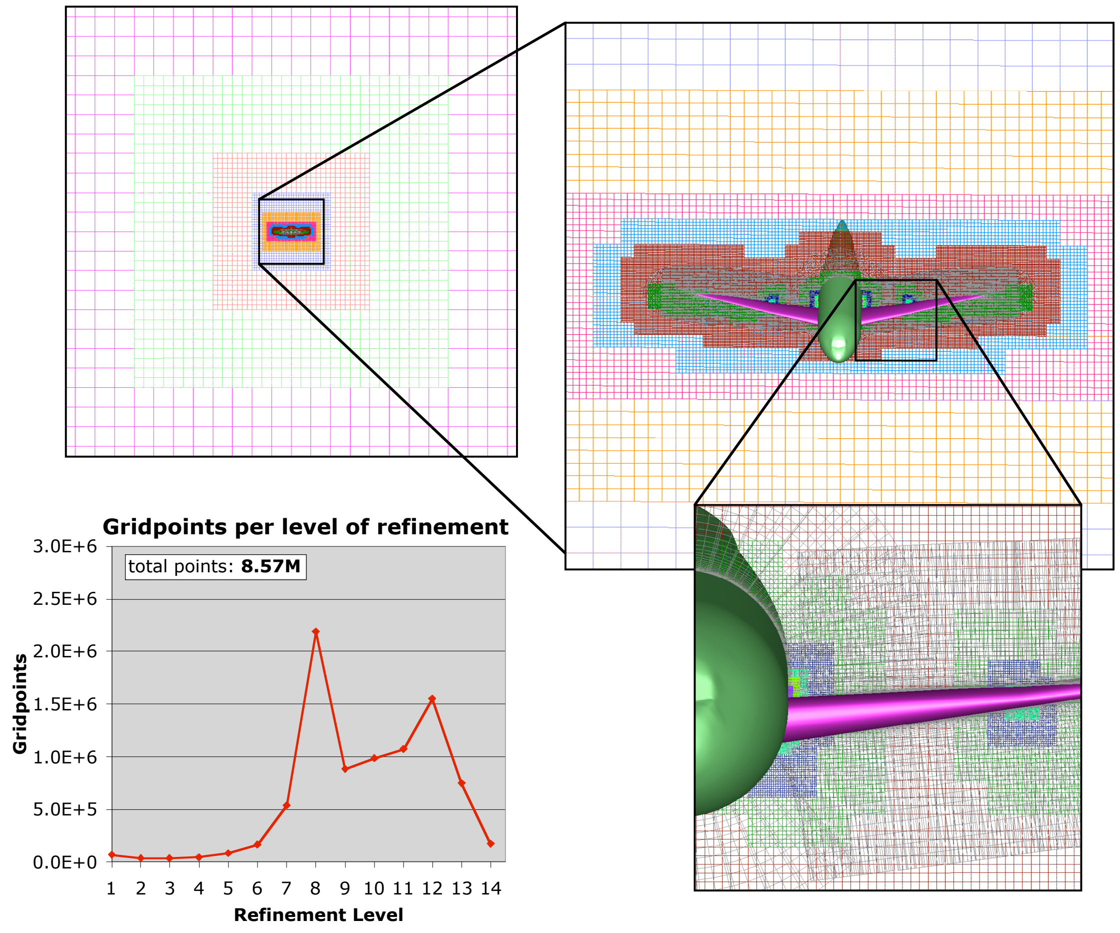


Figure 11. Block-Structured Cartesian AMR grid for generic wing-body strand grid. 14 Levels of refinement, refine ratio of 2. Note that maximum strand spacing is comparable to AMR level 8.

E. Off-Body Flow Solver

A high-order flow solver, ARC3D, that fully exploits the advantages inherent to block-structured uniform Cartesian grids is applied in off-body domains. ARC3D is a new solver that is 6th order accurate in space with a seven-point central difference operator, and 3rd order accurate in time with a three-stage explicit Runge-Kutta time integration scheme. The solver is made parallel and adaptive using the SAMRAI package from Lawrence Livermore National Laboratory.[5] The SAMRAI controlled application of the ARC3D solver is hereafter referred to as SAMARC.

SAMRAI adaptively regenerates the patches to refine to time dependent features or moving strand grids. Each adaptation cycle involves constructing new levels and distributing them to processors. The initial grid generation steps are done in parallel by the CPU owning the old patch; once the patch layout for the new level is known, they are re-distributed to processors in a load balanced way, and data is transferred from the old to the new set of patches. Specifically, the adaptive gridding algorithm proceeds using a four-step process:

i. Choose cells on the next coarser level that require refinement (i.e., cell-tagging).
ii. Construct new patch regions on the finer level from tagged cells.
iii. Move data from the old finer level to the newly generated one.
iv. Generate the new data dependency information required to exchange data at patch boundaries.

Between adaptive gridding steps the governing equations are numerically integrated on the patches using the RK algorithm. Before each RK sub-step, data is exchanged to update patch boundary information between processors using MPI communication operations. In particular, solution data is communicated in a coarse-to-fine cascade to satisfy boundary information on the finer levels. Serial numerical operations that perform an explicit time advance are then applied simultaneously on each patch. At the end of the RK sub-step, a fine-to-coarse communication cascade is executed by injecting fine grid solution data to the coarse levels. Parallel communication for all these operations is entirely managed by SAMRAI.

Figure 12 shows the adaptive solution of a moving Lamb vortex performed with SAMARC. This case uses 6th order spatial differencing because the higher order scheme does a better job at resolving vortex propagation. The solution compares favorably to a uniform grid solution with the same resolution, supporting the notion of AMR effectiveness in terms of solution accuracy as well as computational efficiency. ARC3D also has the option of using 2nd or 4th order schemes as well, for cases that may be more appropriate for lower-order schemes. This is an important point here, since the present near-body strand solver is only 2nd order accurate in space.

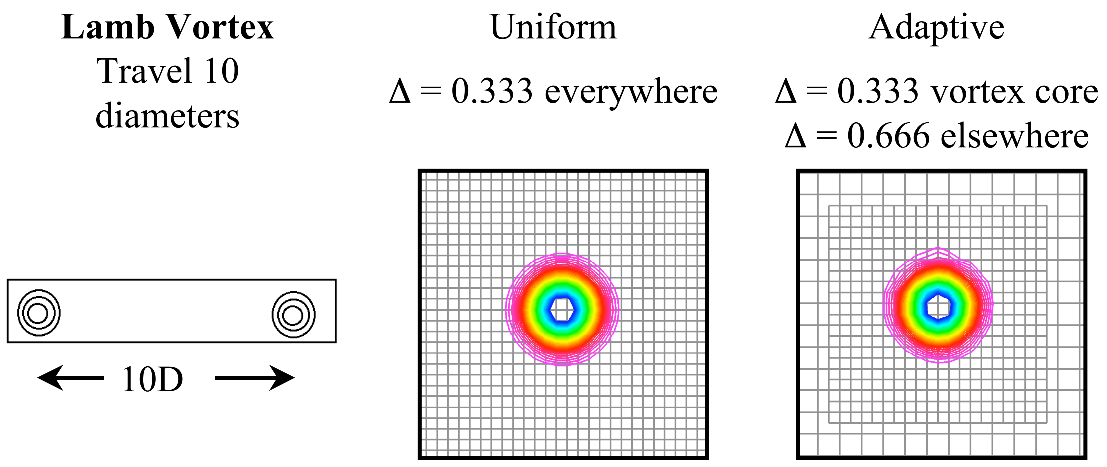


Figure 12. Block-Structured Cartesian AMR Solver for off-body grid system. Lamb vortex propagation with 6th order spatial differencing and 10 points across vortex core.

F. Domain Connectivity

The discretization paradigm proposed here is inherently an overset approach and derives significant advantages thereby. Specifically, the ability to address general multiple-body applications that may involve arbitrary relative motion between bodies, or body components, is possible because overset technology accommodates independently gridded components. The cost of the advantages inherent to such an approach is reflected in the need to establish domain connectivity. Domain connectivity is the communication of flow solution variables between grid components.[12] A number of software products have been developed for this purpose for both structured and unstructured grid systems (see, for example, Refs. 13-17, among others). Although considerable advances have been made in this discipline, next to the task of preparing surface grids suitable for flow simulation modeling from CAD, establishing domain connectivity for complicated applications remains the one that requires the highest degree of user expertise and man-hours. Accordingly, relevance in a design environment depends significantly on an ability to develop fully automatic and robust methods to perform this task. Applications that involve relative motion require domain connectivity to be re-established at every time-step, so computational efficiency is also critical. The most efficient domain connectivity software available today that accommodates relative motion, costs 10 to 20 percent of a flow solver iteration.

```
STAGE-1 SSDC

    BEGIN PARALLEL LOOP (on all procs, NP = 1,NPROCS)
        1.  Perform Near-Body SSDC for the strand-points $P_n$ assigned to proc NP
            a)  ID all positive volume strand elements E that contain $P_n$
            b)  Select optimal donor of set for $P_n$
        2.  Set clipping-index for all strands assigned to proc NP
    END PARALLEL LOOP

Off-Body Grid Generation
    BEGIN PARALLEL SAMRAI
        IF (Off-Body grid system does not exist)
            use STAGE-1 SSDC clipping-indices to create
        IF (ADAPT)
            use STAGE-1 SSDC clipping-indices for geometry based refinement
            and solution error estimates for off-body refinement
    END PARALLEL SAMRAI

STAGE-2 SSDC

    BEGIN PARALLEL LOOP (on all procs, NP = 1,NPROCS)
        1.  Perform Off-Body SSDC for strand points $P_n$ assigned to proc NP
            a)  ID finest resolution off-body grid component that bounds $P_n$
            b)  Given $P_n$, compute bounding element, E on Cartesian grid
            c)  If E spacing < $P_n$ strand spacing, use E as donor
        2.  Finalize clipping-index for strands assigned to proc NP
        3.  ID "terminal-level" off-body grids assigned to proc NP
        4.  Perform Near-Body SSDC for all "terminal-level" points
            assigned to proc NP
    END PARALLEL LOOP
```

Figure 13. Domain Connectivity pseudo-code.

The use of near-body strand grids in combination with off-body block-structured AMR has very significant advantages with respect to domain connectivity. As noted earlier, very little memory is required for grid related data of any type in this paradigm. On parallel compute platforms, this means that all processors have all grid related data resident in local memory. Accordingly, any given processor can locally satisfy all domain connectivity needs relating to the problem component assigned to it. This attribute is referred to hereafter as Self-Satisfying Domain Connectivity, or SSDC. Further, the respective near-body and off-body data structures trivialize interpolation donor identification. Donor elements for points inside the near-body domain can be robustly identified in at most *N* queries, where *N* is the number of near-body surface elements. In practice, the cost is much less due to "chaining" and gradient search[12] operations. Chaining is simply the practice of choosing the donor solution identified for the previous point on a strand as the initial guess at the solution for the current point. Assuming that donor recipients are considered in a strand root-to-tip order, chaining yields the correct answer to within 1 element virtually every time. Donor elements for points that lie in the off-body domain are identified by a trivial direct computation at a cost of 2 adds and 1 multiply. The pseudo-code given in Figure 13 outlines the tasks required to establish domain connectivity for strand grid systems combined with off-body block-structured AMR. Domain connectivity is established in two stages. Stage-1 establishes the domain connectivity between overlapping near-body strand grid components. The clipped points that do not have donors after this stage provide the information needed to generate the off-body grid system. Then, Stage-2 establishes connectivity between near-body strand and off-body Cartesian grid components. Both stages can be performed entirely in parallel.

Subject to conditions of existence, Stage-1 SSDC identifies the optimal near-body solution donor element for all points associated with strands assigned to the local processor, and sets the clipping-index for each strand. The clipping-index is set one point nearer the root than the first recipient point identified on strands for which donor(s) exist, or to the tip minus a minimum overlap parameter for strands with no donors. The minimum overlap parameter, NFRINGE, is user specified, but defaulted to 2 for conventional 2nd order spatially accurate flow solvers. The strand grid generator described in Section II-B sets the clipping-index for strands associated with negative prism elements. Traversing strands in a root-to-tip order, the clipping-index is set one point ahead of the first negative volume element identified. Strand grids require only the simplest definition of optimality – the donor element with the lowest strand index (finest spacing, which is nearest the root).

Given near-body strand grid definitions, clipping-indices resulting from a Stage-1 SSDC step, and the extent of the far-field domain, the SAMRAI module described in Section II-D automatically (and in parallel) generates the complete off-body system of grids, telescoping as needed to provide resolution compatible with the clipped strands. As indicated in the pseudo-code of Figure 13, this is exactly what occurs at start-up. However, if the domain connectivity step coincides with a solution adaptation cycle, SAMRAI uses estimates of solution error, or feature detection information in combination with the information noted above, to automatically generate the off-body grid system.

All domain connectivity requirements not satisfied by Stage-1 SSDC (i.e., clipped strands that are not overlapped by other strand prisms) are finalized in the Stage-2 SSDC step indicated in Figure 13. The Stage-2 SSDC step is a parallel operation that completes both a) off- to near-body and b) near- to off-body paths of communication for all grids allocated to a given processor. The off- to near-body path is established by execution of Stage-2 SSDC steps 1 and 2. Here, optimal donors are computed from the terminal (finest refinement) level off-body grids that overlap each near-body strand. Again, "optimal" simply means that donors are accepted along the strand for all strand points bounded by a terminal level off-body Cartesian grid element of finer resolution capacity than is native to the strand position of the point in question. The reverse communication path (near- to off-body) is then established by execution of Stage-2 SSDC steps 3 and 4. Recall that the hierarchical off-body grid system, multiply covers regions of the domain with both coarse and fine levels. However, it is only necessary to communicate the near-body solution to the finest level grids that overlap the set of near-body prism stacks. Accordingly, all terminal level off-body grid points bounded by prism elements of finer resolution, receive corresponding solution information from the respective prism elements. The domain connectivity task here is to identify the specific prism elements for the set of terminal level off-body grid points so bounded.

Consider Figure 14 in which a 1-D illustration of the pseudo-code indicated in Figure 13 is given. Figure 14a represents the Stage-1 SSDC step. Since there is no overlap between the strands from the opposing bodies, none of the strand points have donor elements from the opposite strand. Accordingly, the clipping indices are simply set to the respective strand tips minus NFRINGE, as indicated by the red and blue clipping-index symbols. The SAMRAI off-body grid generation step is indicated in Figure 14b. Block-structured AMR adapts to the clipping indices. The resulting uniform Cartesian grid span is extended by the NFRINGE parameter as well, resulting in the black grid indicated. Finally, Figure 14c indicates the Stage-2 SSDC step in which the optimal donor element is identified for all strand points that currently do not have a donor, as well as all points associated with terminal off-body grids that overlap strands. Optimal donors are indicated in the figure as solid circles. The color of the circle indicates the grid providing the donation. The number inscribed in the circle identifies the specific grid index of the element providing the donation. Again, by definition, the optimal donor is the finest element that contains a given point and that has finer resolution than the cell to which the point itself is associated. Accordingly, only points 1, 2, 11, and 12 of the off-body grid have optimal donors, since the off-body resolution is superior to all other strand elements that bound off-body points 3 through 10. Likewise, only points 10, 11, and 12 on each strand have optimal donors, even though the off-body coverage extends further toward the respective strand roots. All strand elements associated with points 1 through 9 have superior resolution to that available in the off-body grid. Note that the clipping indices on the respective strands are adjusted from index 10 to index 9. At the end of a Stage-2 SSDC step, the clipping index is always positioned one point nearer the root than the first optimal donor recipient point on the strand.

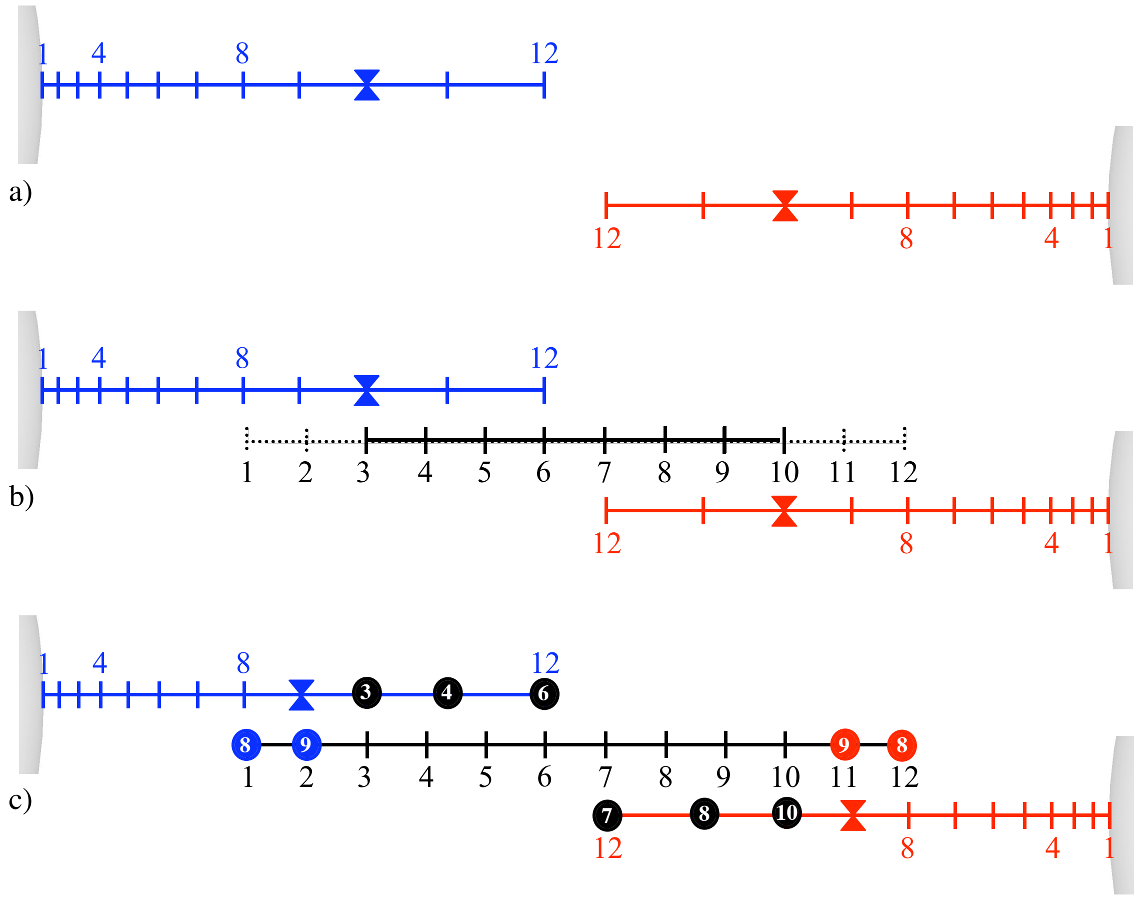


Figure 14. Domain connectivity for opposing strands that do not overlap. a) Stage-1 SSDC. b) Off-body grid generation with SAMRAI. c) Stage-2 SSDC.

Figure 15 is another illustration of the pseudo-code of Figure 13. Here, the opposing bodies are positioned relatively close together such that Stage-1 SSDC yields optimal donors for each strand, leaving no generation requirement for the off-body SAMRAI step. Lacking any other grid and the purely 1-D problem indicated by the figure, there is no work required by the Stage-2 SSDC step either. Here, the clipping index is set to 6 with optimal donors identified for points 7 through 9 on each strand. No donors exist on either strand beyond point 9. Similarly, optimal donors do not exist on either strand inside of point 7, since the receiving strand resolution is as good, or superior to that available on the would-be donor strand. Proper connectivity solutions exist for multiple-body problems where the nearest face of another body is more than NFRINGE points from the opposing roots. Collisions[18] result when bodies advance closer than this.

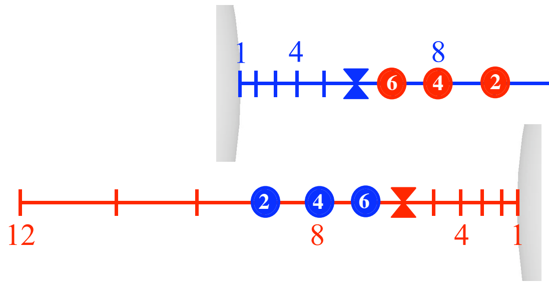


Figure 15. Domain connectivity solution for closely positioned bodies. Result from Stage-1 SSDC.

In a "chimera" sense, the approach described above *implicitly*[15] defines "holes," in that holes are a by-product of the optimal donor identification step. Point types along all strands adhere to the following protocol: (1) root, field, fringe, or (2) root, fringe. There are no other possibilities, or exceptions. Field points are all points along a strand between the root and clipping index, inclusive. Fringe points are strand points that are recipients of optimal donor interpolation information. All strand points that are more than NFRINGE points beyond the clipping-index are effectively chimera holes. This concept is evident in Figure 15. This convention holds as well for strands that protrude inside a neighboring body and is illustrated in Figure 15. Another example of implicit holes is given in Figure 16. Here, the relative position of the bodies and associated strands result in all points beyond point 6 in both the blue and red strands to be considered hole-points. This is so even though donors exist for most points in these intervals (from the purple strand for blue points beyond point 7, and from the aqua strand for the red points beyond point 7).

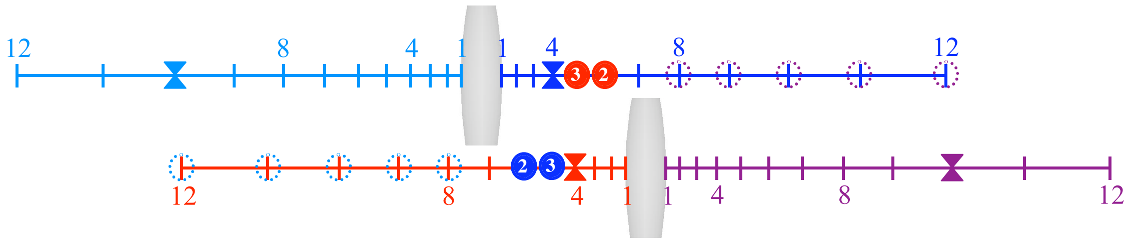


Figure 16. Strand protocol. Stage-1 SSDC marking of 4 strands.

Near-body strand grids in combination with off-body block-structured Cartesian AMR is a powerful combination that holds real potential for fully automatic domain connectivity that is highly efficient, parallel, and scalable. Presently, the method is in early stages of development. Accordingly, much testing and additional developmental tasks remain. The preliminary integration of these ideas, solver, off-body AMR, domain connectivity, etc., is demonstrated in the next section. The approach is intended for far more complex situations than are given here. Still, this is the beginning.

III. Applications

The strand grid approach is tested for a standard problem in CFD, flow over a sphere. Although this problem is quite simple geometrically and does not highlight the advantages of the automated complex geometry and grid attributes inherent to the approach, it is a useful test for three reasons. First, it is a well-known problem with documented results, both computed and experimental. Second, it is a good test to analyze the data transfer between two solvers (near-body and off-body). Although the present solvers (NSU3D and SAMARC) have been analyzed thoroughly in other works, interfacing the two is a key novel feature of the strand grid approach and requires verification. Third, it is an excellent baseline to measure desired improvements in computational efficiency, both memory and execution, of a future near-body solver designed to fully exploit the advantages inherent to strand grids. All cases described in this section were computed on 16 processors of an Opteron-based Linux cluster.

Recall that definition of a strand grid requires only coordinates of the surface vertices, a strand direction vector for each vertex, and a single strand distribution function. In the present case, the sphere surface is discretized with 1,996 triangles. Since the radius of curvature for a sphere is constant, strand direction vectors do not require smoothing and are simply unit surface normal vectors. A hyperbolic tangent stretching is used here to define the strand distribution function, ranging from a y^+ value of 1 at the root and transitional spacing at the tip, with a total of 21 points. Accordingly, the present near-body strand grid has a total of 39,920 prism cells.

An off-body block-structured uniform Cartesian grid system is automatically generated to cover the far-field. The overall extent of the problem domain and an initial coarse-grid spacing is the only required SAMARC input. Given this, the off-body grid system is automatically generated, adapting to match the transitional spacing available in the near-body strand grid. In this case, the off-body grid extends 7 diameters upstream (X) and to the sides (Y,Z), and 14 diameters downstream. SAMARC input for the coarse grid spacing is set such that the Cartesian adaptive grids match the near-body transitional spacing after five levels of refinement (see Fig. 17). Away from the body, the Cartesian grids adapt to regions of high vorticity. The finest spacing away from the body is restricted to be five levels as well, although it is possible to use finer spacing away from the body if desired. The adaptive SAMARC grid ranges between 2.5M and 13M points as it adapts to regions of high vorticity.

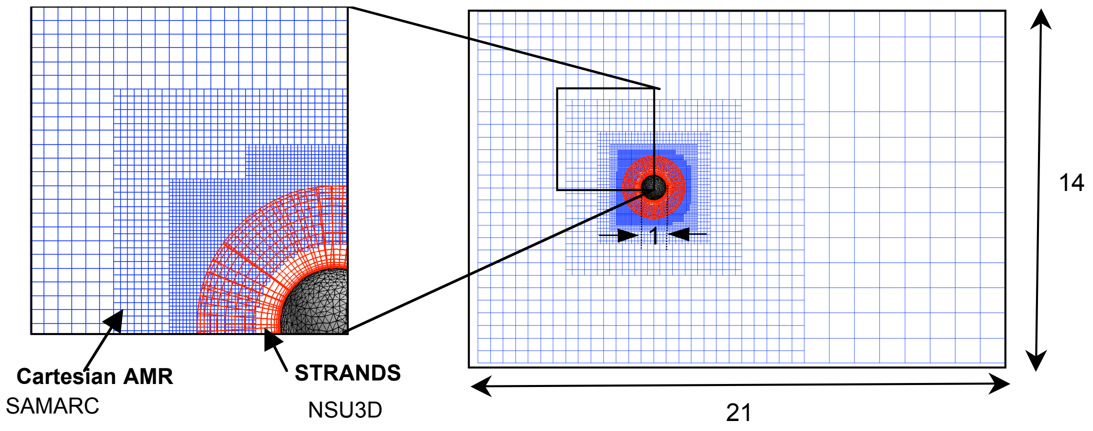


Figure 17. Near-body/Off-body grid system for sphere at the initial time.

Flow conditions behind the sphere change with Reynolds number - Re=UD/ν. Linear stability analysis as well as various computational and experimental results show that downstream flow remains steady and axisymmetric for Re < 210, and becomes unsteady and demonstrates periodic shedding at Re > 290. Figure 18 shows the computed recirculation pattern in the steady regime at Re=120. Note that the recirculation zone extends between the prismatic and Cartesian grid systems. To verify the accuracy of the recirculation characteristics, the separation angle, diameter, and center location of the recirculation bubbles are extracted from the present simulation results for a Reynolds number sweep from 40 to 160. Figure 19 compares the present simulation results to the computed results of Magnaudet[19] and Tomboulides[20], and to the experimental results of Pruppacher[21] and Taneda.[22] The present solutions generally show very good agreement with the reference experimental and computational results. An ability to correctly predict unsteady shedding at higher Reynolds number is illustrated in Figure 20. Here, results are shown for Re=300, with dynamic grid refinement keyed to vorticity magnitude in the off-body Cartesian grids. Periodic shedding is clear.

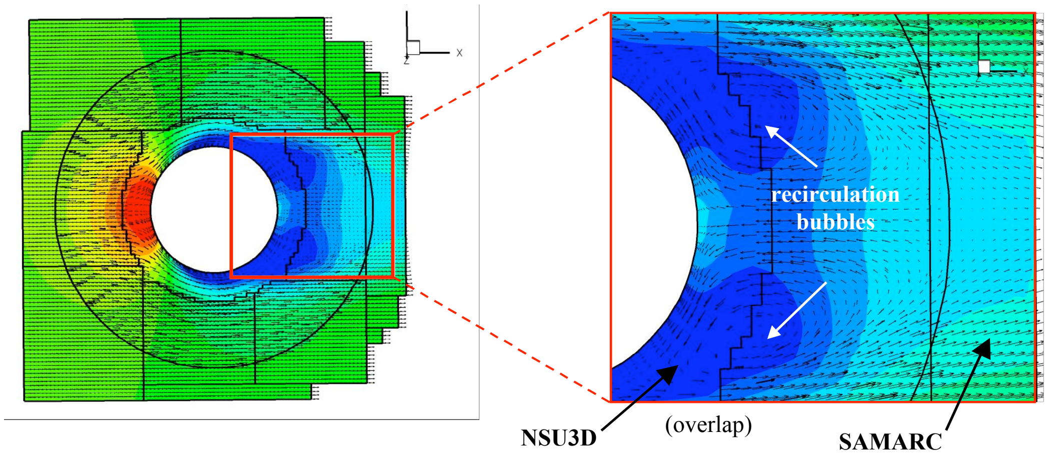
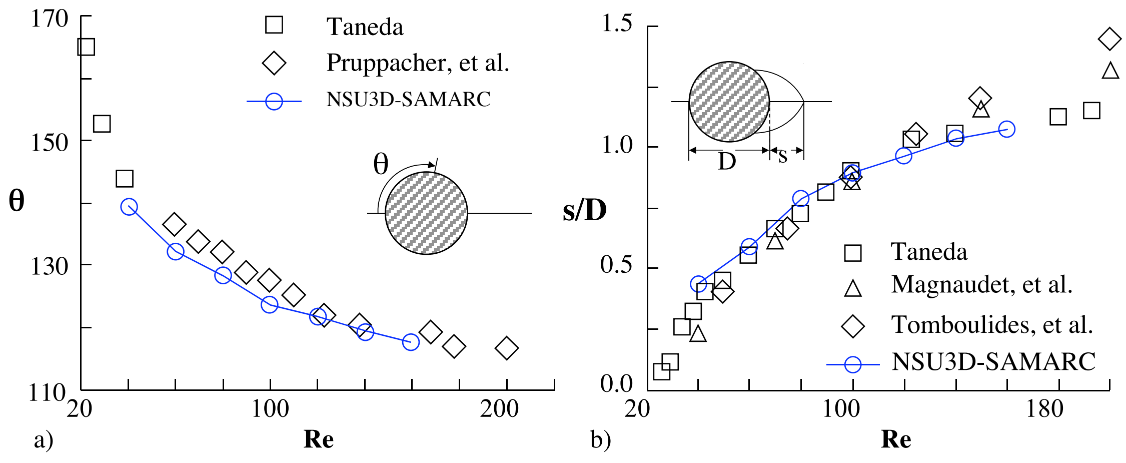


Figure 18. Recirculation bubbles behind sphere at Re=120.



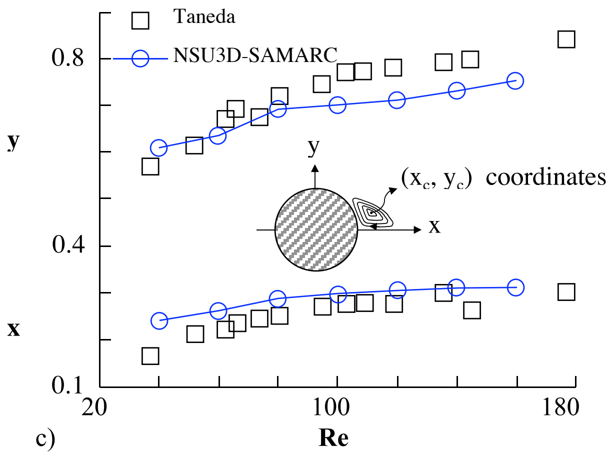


Figure 19. Steady-state flow over sphere (Re = 40 to 160). a) Separation angle, b) Recirculation bubble size, and c) Bubble center location.

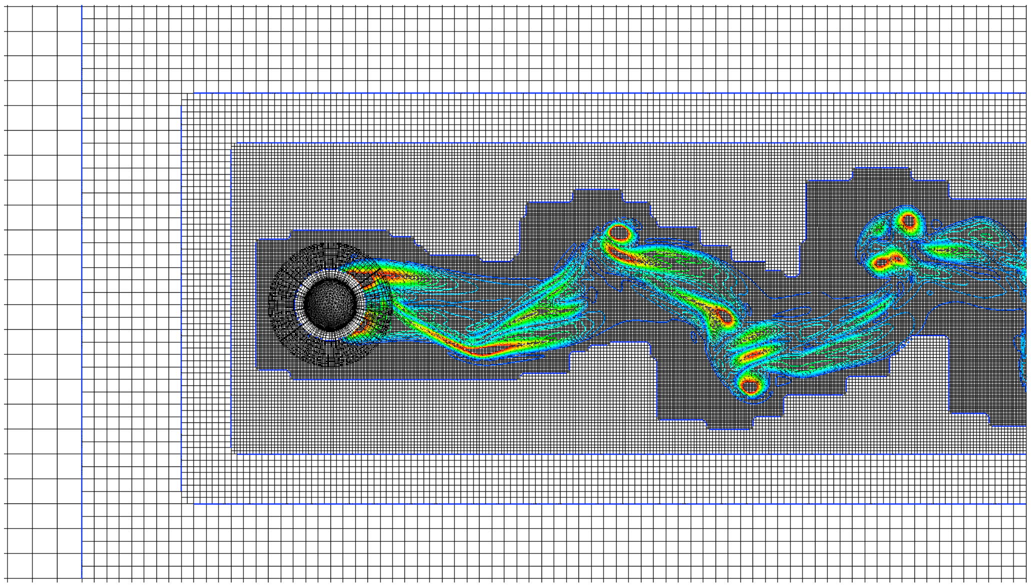


Figure 20. Vorticity contours for unsteady flow at Re=300. Grids adapted to regions of high vorticity.

IV. Concluding Remarks

Strand grid technology is intended for applications involving internal/external flows, steady/unsteady flows; single or multiple body applications with static/dynamic body geometry, rigid/elastic body motion; and all practical combinations thereof. Although the technology is presently immature, initial integration of all algorithm functionality described in Section II has been completed. The present article establishes the correctness and baseline performance capability of the present implementation and suggests the general feasibility of the approach.

Further, the potential of strand grids for robust and automatic treatment of dynamic complex geometry, suggests that it may be possible to expand the spectrum of solvable problems to include such things as grossly deformable and fragmenting bodies. Certainly, there are aspects to the physics of these problem classes that require technology beyond that necessary to deal with geometric complexities. Accurate prediction of fracture lines for fragmenting bodies, for example, is a discipline unto itself. Still, strand grid technology may allow initial forays into problem classes that have as yet been closed to closely coupled high fidelity multi-physics simulation. Aircraft canopy ejection and control surface movements (e.g., flap deployment) are important preliminary “fragmentation” situations that have been problematic thus far using existing HPC tools.

Acknowledgements

The authors gratefully acknowledge the support of the U.S. Department of Defense High Performance Computing Modernization Office for support of this work through the HPC Institute for Advanced Rotorcraft Modeling and Simulation (HI-ARMS). Also, thanks are due to our host organizations, the U.S. Army Aeroflightdynamics Directorate, NASA Ames Research Center, the ELORET Corporation, and the National Institute of Aerospace. The authors express special thanks to Dr. Thomas Pulliam, who wrote the ARC3D solver discussed in section II-E, and to Prof. Dimitri Mavriplis, who wrote the NSU3D solver noted in section II-C.

References

1 Aftosmis, M. J., Berger, M. J., Melton, J. E., “Robust and Efficient Cartesian Mesh Generation for Component-Based Geometry,” AIAA 97-0196, 35th Aerospace Sciences Meeting, Reno, NV, Jan. 1997.

2 Aftosmis, M. J., Berger, M. J., and Adomavicius, G., “A Parallel Multilevel Method for Adaptively Refined Cartesian Grids with Embedded Boundaries,” AIAA-2000-0808, 38th Aerospace Sciences Meeting, Reno, NV, Jan. 2000.

3 Landsberg, S. M. and Boris, J. P., “The Virtual Cell Embedding Method: A Simple Approach for Gridding Complex Geoemtries,” AIAA-97-1982, 13th AIAA Computational Fluid Dynamics Conference, Snowmass Village, CO, June 1997.

4 Meakin, R., “An Efficient Means of Adaptive Refinement for Overset Structured Grids,” AIAA-95-1722, 12th AIAA Computational Fluid Dynamics Conference, San Diego, CA, June 1995.

5 Hornung, R.D., Wissink, A. M., and Kohn, S. R., “Managing Complex data and geometry in parallel Structured AMR Applications,” Engineering with Computers, 2006. (to appear) (http://ww.llnl.gov/CASC/SAMRAI)

6 Meakin, R., “Adaptive Spatial Partitioning and Refinement for Overset Structured Grids,” Comput. Methods Appl. Mech. Engrg. Vol. 189, No. 4, Sept. 2000, pp. 1077-1117.

7 Haimes, R., and Aftosmis, M. J., ``On Generating High Quality "Water-tight" Triangulations Directly from CAD," Meeting of the International Society for Grid Generation, (ISGG), Hawaii, June 2002.

8 Haimes, R., and Aftosmis, M. J., ``Watertight Anisotropic Surface Meshing Using Quadrilateral Patches," 13th International Meshing Roundtable, Sept. 2004.

9 Marcum, D. L., ``Efficient Generation of High-Quality Unstructured Surface and Volume Grids," Engineering with Computers, Vol. 17, No. 3, Oct. 2001.

10 Mavriplis, D. J. and Pirzadeh, S., "Large-Scale Parallel Unstructured Mesh Computations for 3D High-Lift Analysis," AIAA J. of Aircraft, Vol. 36, No. 6, Dec. 1999, pp. 987-998.

11 Berger, M. J. and Colella, P., “Local Adaptive Mesh Refinement for Shock Hydrodynamics,” Journal of Computational Physics, 82:64-84, 1989.

12 Meakin, R., “Composite Overset Structured Grids,” *Handbook of Grid Generation*, edited by J. Thompson, B. Soni, and N. Weatherill, CRC Press, Boca Raton, 1999, Chapter 11.

[13] Henshaw, W., "Ogen: an Overlapping Grid Generator for Overture," Research Report UCRL-MA-132237, Lawrence Livermore National Laboratory, 1998. (http://www.llnl.gov/CASC/Overture/henshaw/documentation/ogen.pdf)

[14] Meakin, R. L., "Object X-rays for Cutting Holes in Composite Overset Structured Grids," AIAA-2001-2537, 15th AIAA Computational Fluid Dynamics Conference, Anaheim, CA, June 2001.

[15] Lee, Y-L. and Baeder, J. D., "Implicit Hole Cutting – A New Approach to Overset Grid Connectivity," AIAA-2003-4128, 16th AIAA Computational Fluid Dynamics Conference, Orlando, FL, June 2003.

[16] Rogers, S. E., Suhs, N. E., and Dietz, W. E., "PEGASUS 5: An Automated Pre-processor for Overset-Grid CFD," AIAA J., Vol 41, No. 6, June 2003, pp. 1037-1045.

[17] Noack, R.W., "SUGGAR: a General Capability for Moving Body Overset Grid Assembly", AIAA-2005-5117, 17th AIAA Computational Fluid Dynamics Conference, Toronto, Ontario Canada, June 2005.

[18] Meakin, R. L., "Multiple-Body Proximate-Flight Simulation Methods," AIAA-2005-4621, 17th AIAA Computational Fluid Dynamics Conference, Toronto, Ontario Canada, June 2005.

[19] Magnaudet, J., M. Riviero, and J. Fabre, "Accelerated flows past a sphere or spherical bubble. Part 1: Steady streaming flow," *J. Fluid Mechanics*, 284, pp. 97-135, 1995.

[20] Tomboulides, A.G., S. A. Orszag, and G. Karniadakis, "Direct and Large Eddy Simulations of axisymmetric wakes," AIAA Paper 93-0546, 1993.

[21] Pruppacher, H.R., B. P. Le Clair, and A. E. Hamielec, "Some relations between drag and flow pattern of viscous flow past a sphere and cylinder at low and intermediate Reynolds numbers," *J. Fluid Mechanics*, 44, part 4., pp. 781-791, 1970.

[22] Taneda, S. "Experimental investigation of wake behind a sphere at low Reynolds numbers," *J. Physical Society of Japan*, 11, No. 10, pp. 1104-1108, 1956.